Ensembling Methods Applied to Deep Learning Approaches for Twitter Sentiment Analysis

Ethan Swistak
Ulm University
ethan.swistak@uni-ulm.de

Mohammed Sabiya Sujith Ahamed
Ulm University
mohammed.sujith-ahamed@uni-ulm.de

Nikhil Betgov
Ulm University
nikhil.betgov@uni-ulm.de

Tushar Singhal
Ulm University
tushar.singhal@uni-ulm.de

Abstract

Sentiment analysis is a study of identifying people's emotion and attitude in very short messages such as in the movie review, stock exchanges, social networking and so on. To classify the text, traditional machine learning algorithms have been used in the early years. But the accuracy is not the best due to its limited capability to efficiently classify a large amount of data. In this paper, we have classified the Twitter tweets based on three sentiments such as Neutral, Positive, and Negative. We have used GloVe embedding for the feature extraction of the text. This study proposes two deep learning algorithms, namely Convolutional Neural Networks (CNN) and Long-Short Term Memory (LSTM), to classify the tweets. The important topic of this research is to analyze whether the ensembling methods boost the performance of the strong classifiers. So, We have trained 5 CNN models and 5 LSTM models for ensembling. We have compared two ensembles, namely stacking prediction and stacking generalization. This research resulted in a small performance boost with the stacking prediction and a smaller with the stacking generalization in terms of F1-score.

Keywords

Sentiment Analysis, GloVe Embedding, Ensembling, Stacking Predictions, Stacking Generalization, F1-score.

1 Introduction

The online social networking site Twitter is a popular forum for people to express their opinions on various topics of interest. The key feature is that users can post Tweets, short messages of 280 characters or less, that are publicly visible for others to view and comment on. The simplicity of Twitter, limited character size, and the various feedback mechanisms that users can provide in response to Tweets make it an ideal target for a variety of machine learning applications, in particular, gauging user sentiment. The problem of sentiment analysis can be formalized as Given a character string and a ground truth labelling of whether such a string is considered positive or negative, predict whether other similar character strings are positive or negative. Although this classification task outputs only a simple binary classification, the vastness of the input space and the number of different orthographic and morphological differences has made traditional approaches to sentiment classification prohibitive.

Traditional Natural Language Processing (NLP) techniques and Multilayer Perceptron (MLP) are well-known methods to solve sentiment classifications. However, the revolution of Deep Learning made the breakthrough in many application domains, more recently in NLP. In comparison, Deep Neural Networks (DNN) have shown better accuracy and efficiency. In this research paper, we propose comparing two different deep learning approaches to gauging sentiment: Long-Short Term Memory networks (LSTM) and Convolutional Neural Networks (CNN). LSTM networks utilize a series of memory cells that are able to modulate their activation based on past contextual information passed forward from previous iterations of the network. They have been applied successfully to many NLP tasks in the past and are a good candidate for solving sentiment classification problems. The latter network architecture, CNN, has generally found applications in the areas of computer vision but it has been discovered that they also perform well in a variety of NLP contexts given their ability to recognize local structure in a vectorized representation of an utterance.

The main contribution of this research is to compare these two deep network architectures' performance on sentiment analysis and to study whether the ensemble of CNN and LSTM gives a better classification result. This paper is based on the SemEval-2017 research paper, Twitter Sentiment Analysis with CNNs and LSTM by Mathien Cliche [2].

This paper has been organized according to the following structure. Section 2 presents the related work, the experimental design will be introduced in Section 3, the result will be presented in Section 4, and the discussion and conclusion will be in Section 5 and Section 6.

2 Related Work

Sentiment analysis has been proposed and researched over the years. In its early stage, it uses binary classification and ANN to classify the sentiment values. More recently, several machine learning and deep learning techniques are used to classify the texts on Twitter. Yujie Lu et al [4]. proposed a Support Vector Machine (SVM) using unigram and bigram features as a baseline model and two deep learning models, CNN and LSTM, to classify the tweets. He

This is a Master degree project in Ulm University submitted by Ethan Swistak, Mohammed Sabiya Sujith Ahamaed, Nikhil Betgov, and Tushar Singhal, to the project supervisor Prof. Dr. Ansgar Scherp, email: ansgar.scherp@uni-ulm.de, Department of Databases and Information Systems (DBIS), Ulm University, 89081 Ulm, Germany.
Project Report, March 2021, Ulm University, Ulm, Germany

compared binary classification (Positive and Negative) and three-class classification (Positive, Negative, and Neutral). It resulted from that SVM with unigram and bigram features gave the highest accuracy in both binary classification and three-class classification.

Heikal et al [3]. used Arabic tweets for sentiment analysis. He used CNN and LSTM to predict the sentiment of Arabic tweets. It uses AraVec for word embeddings which were pre-trained on Twitter data following Word2vec. Despite the complexity of the Arabic language and the simplicity of the system used, the ensemble model achieved an F1-score of 64.46%

Muhammad et al [7]. proposed a combination of CNN-LSTM deep networks for the sentiment classification on Twitter datasets. For Feature extraction, he used two methods, term frequency-inverse document frequency and word2vec. He compared the deep learning classifiers with some other machine learning classifiers such as Support Vector Machine (SVM), Random Forest (RF), Stochastic Gradient Descent (SGD), Logistic Regressions, and Voting classifiers. It demonstrated that CNN-LSTM yielded higher accuracy.

Stojanovski et al [6]. presented both convolutional and gated recurrent neural networks to observe the tweet representations. The networks are trained on the pre-trained GloVe word embeddings. The CNN has experimented with multiple filters with additional window sizes of 4 and 5 with a fully connected softmax layer and dropout regularization. It is evaluated on the Semval 2016 datasets, where the proposed model achieved an Average F1 score higher than Baseline scores.

Pedro et al [5]. also proposed the combination of CNN-LSTM models for the binary classification on Twitter datasets. It also compared the pre-trained and non-pre-trained word embeddings. The results showed that the non-pre-trained embeddings give higher accuracy than pre-trained. The paper discussed the effects of using dropouts in the neural network models and the effects of different epochs while training the model.

3 Experimental Design

3.1 Data Acquisition

Datasets: We have used the SemEval2017 dataset initially for our experiments. Due to insufficient and imbalanced data, a combined corpus of SemEval2017, Airlines dataset and a sampled subset of the Sentiment140 dataset has been used for our work.

3.2 Preprocessing

Before we actually work on the tweets, we preprocess them using various techniques. As explained in the previous section, a combined corpus of the SemEval2017, Airlines and Sentiment140(few sampled tweets to combat data imbalance for the neutral class) has been used as our final mega dataset to carry out our experiments. The counts of the individual classes of the tweets are:

- Neutral: 25348
- Positive: 47028
- Negative: 41842

As we see, there is still a data imbalance with regard to the neutral class, but not to a great extent. We follow the methods as proposed by [1] with some minor changes:

- All URLs found in the tweets are removed as they do not contribute toward the sentiments of the tweets.
- All tweets are converted to lowercase.
- All twitter handles and twitter return handles are removed as they do not contribute toward the sentiments of the tweets.
- Extra spaces in tweets are removed.
- Special characters are not removed(reasons for this are discussed in a later section).
- Stop words are not removed(reasons for this are discussed in a later section).

3.3 Metrics

The metrics used for evaluating our models are Accuracy, Precision, Recall and F1. We emphasize the F1 metrics while interpreting our results. This is due to the fact that we have a small imbalance in data and so we would like to see the harmonic mean of the precision and recall metrics, which is exactly what the F1-score indicates. The metrics used by [1] in the results are also F1 and this serves as a baseline for our comparison as well.

3.4 Architectures

Convolutional Neural Network (CNN):. We use a variation of a CNN that we call a Flat CNN which uses a variety of different convolution sizes on a single layer. We found that in addition to validation results being slightly higher for this version of the CNN, it was also a very simple model to train.

Flat CNN utilizes a set of three different convolution sizes on a single layer. The width of the convolutions are fixed at the length of the individual word vectors and the height determines how many words the CNN will examine at a time. In this way, we can factor in different word lengths and hopefully achieve a performance boost for ensembles. We tried 2, 8, and 16 filters per convolution and found that eight was optimal in providing the network with sufficient capacity to classify the tweets without overfitting.

We believe that the current architecture achieves superior results because it is better structured to prevent learning spurious correlations between words and overfitting to the training data. In addition, it uses an order of magnitude fewer parameters compared to a traditional CNN and processes the text in a much more sequential fashion similar to human reading.

Long-Short Term Memory (LSTM): Long Short Term Memory networks are best used when sequence data is involved. They belong to the family of RNNs(Recurrent Neural Networks) which attempt to share the weights along the input sequence. LSTMs solve the problems of RNNs, namely: exploding and vanishing gradients, long term dependencies. These problems are solved by the memory cells in the LSTM units which basically hold the information for a longer period of time.

The cell state of an LSTM which consists of the Forget gate, input gate and output gate, controls how the information enters the

memory cell, when it is kept or forgotten and when it exits through the output. LSTMs also have hidden states which are nothing but the outputs.

According to [2], a simple LSTM uses the current word embedding and the previous hidden state to compute the new hidden state, and this is done for every word in the sequence. The hidden state at time t, is computed by [8]:

$$f_t = \sigma(\mathcal{W}_f \cdot x_t + \mathcal{U}_f \cdot h_{t-1} + b_f) \tag{1}$$

$$i_t = \sigma(\mathcal{W}_i \cdot x_t + \mathcal{U}_i \cdot h_{t-1} + b_i) \tag{2}$$

$$o_t = \sigma(\mathcal{W}_o \cdot x_t + \mathcal{U}_o \cdot h_{t-1} + b_o) \tag{3}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ tanh(\mathcal{W}_c \cdot x_t + \mathcal{U}_c \cdot h_{t-1} + b_c) \tag{4}$$

$$h_t = o_t \circ tanh(c_t) \tag{5}$$

where,

$i_t = input\ gate$

$f_t = forget\ gate$

$c_t = cell\ state$

$h_t = hidden\ state$

$\sigma = sigmoid\ function$

$\circ = hadamard\ product$

$\mathcal{W}_f, \mathcal{W}_i, \mathcal{W}_o$ *and* $\mathcal{U}_f, \mathcal{U}_i, \mathcal{U}_o = weight\ matrices$

$b_f, b_i, b_o, b_c = bias\ terms$

The authors of [2] rightly point out that this simple LSTM still has one problem and that is it doesn't take into account post word information as the processing of the sequences is done only in the forward direction. In order to combat this issue, a more superior version of the LSTM known as the bidirectional LSTM is used. In this architecture, one LSTM processes the input in the forward direction and the other in the reverse direction. In our work too, we utilize the bidirectional LSTMs with changes(discussed in a later section) to the various overall LSTM models we build and find that these models are very good at handling sequence data.

3.5 Procedure

For our study, we decided to train individual CNN and LSTM models and then attempt different methods of ensembling the models to determine if better results could be achieved through a combined model than anyone individually. After some parameter tuning, we came to a preliminary set of architectures and hyperparameters that seemed to generate good results on the validation data.

We start by importing twitter data and associated training labels. In order to reduce variability in the text we eliminate @mentions and only keep raw text. Special characters and stop words were not removed as the models found them important during the learning stage. We considered additionally removing suffixes but this did not appear to improve performance so it was not used in the final iteration.

3.5.1 GloVe Embeddings: Glove stands for Global Vectors for Word Representation and is an algorithm for building a vectorized representation of a text corpus off of a co-occurrence matrix. Glove, instead of starting with co-occurrence statistics establishes ratios of different co-occurrences with certain "probe words" to establish meaning. For instance, ice appears more commonly next to solid than steam but they both appear with similar frequencies next to water. Each dimension of the word vector space can contain its own independent co-occurrence probability vector and this vector can then be turned into a ratio of the co-occurrences between a certain mediating word. These relationships can be expressed as a 200-dimensional vector for our purposes and fed into a neural network.

We use a Glove Embedding that is already pre-trained on a corpus of 1.6 billion words from Wikipedia, which in our case, helps to arrange the latent space in a way that is more conducive to learning certain word associations. By remapping the input space to take word associations into account, we provide our network with a better starting point for learning to classify the sentiment as similar words with similar sentiments that should be located relatively close to each other.

3.5.2 CNN: We trained five separate CNN's, each with a single convolutional layer consisting of 8 of each of 3 filter sizes, for a total of 24 convolutional filters per network. The filter sizes were [2,3,4], [3,4,5], [5,6,7], [7,8,9], and [9,10,11] words long and the widths encompassed the entire 200 dimensional word vector.

We trained each model with an Adam optimizer for 20 epochs with 3000 tweet mini-batches and additionally added a 25% dropout layer. In addition, we made use of an early stopping and LR reduction callback in order to further optimize the training data. In addition, we only save the best model from each run in an effort to avoid overfitting.

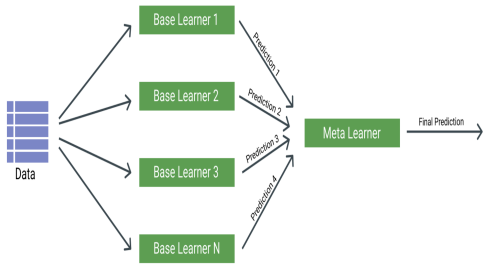
3.5.3 LSTM: Five different LSTM models were built and trained. The hyperparameters that we experimented with are recurrent dropout rate, spatial dropout rate and the number of LSTM units. We also applied dropouts in some models to see if there was any performance shift. Also, we found out that adding dense layers before the last prediction layer resulted in worse performance in contrast to the results of the architecture proposed by [2]. Overall,

keeping the spatial dropout, recurrent dropout and dropout in the range of 0.3-0.5 and the number of LSTM units close to 200 resulted in optimal performance.

All LSTM models were optimized using the Adam optimizer and trained for 20 epochs with a batch size of 3000 tweets. In an attempt to minimize and avoid overfitting, we used an early stopping and learning rate reduction callback. Only the best models were saved.

3.5.4 Ensembling: Multiple models are combined using ensemble methods to boost the prediction of the individual models. The traditional ensemble methods used weak learners. However, the modern approach creates an ensemble model of a well-chosen collection of strong-diverse models. There are different ways to ensemble the classifiers. We used Stacking predictions and Stacking generalization ensemble approaches.

Stacking Predictions: Stacking prediction is an ensemble learning that combines multiple base model predictions as shown in Figure 1, which serves as an input to the second-level Meta classifier. We have performed three forms of stacking predictions ensembles. First, we have stacked 5-trained Flat CNN models with different filtered sizes. Second, we have stacked 5-trained LSTM with different LSTM units. Then, both the trained Flat CNN models and trained LSTM models are stacked together (10 models). Here, the CNNs and LSTMs are base learners. The meta learner for the ensemble model is a simple model made of multiple dense layers with a dropout of 20%. The ensemble model is trained on 50 epochs and 3000 batch-sizes with early stopping and reduce learning rate callbacks.



Source: Van Nguyen - medium.com

Figure 1: Overview of Stacking Predictions Ensembling.

Stacking Generalization: In Stacking Generalization, as in Figure 2 we combine different models in contrast to combinations of predictions in Stacking Predictions. The output of the models is then connected to a new meta classifier. The weights of the base learners are frozen so that it does not update while training the ensemble model. Here the data has to be duplicated k times before passing it to the ensemble model because the input layer of the base models are used as a separate input head (k is the number of base models) [1]. This model is trained for 20 epochs and 1000 batch size, increasing the batch size gave memory out of bounds issue since the data was already being duplicated k times. All the configurations of base models were the same as stacking predictions.

Table 1: Performance Metrics for Stacking Predictions

Model	Accuracy	Recall	Precision	F1	Boost
CNN	0.6888	0.7973	0.6174	0.6948	0.0291
LSTM	0.7270	0.8209	0.6589	0.7313	0.0047
CNN + LSTM	0.6976	0.8082	0.5593	0.6613	-0.0653

Table 2: Performance Metrics for Stacking Generalization

Model	Accuracy	Recall	Precision	F1	Boost
CNN	0.6874	0.6743	0.6966	0.6851	0.0194
LSTM	0.7275	0.7029	0.7432	0.7221	-0.0045
CNN + LSTM	0.6778	0.6609	0.6924	0.6760	-0.0506

Table 3: Performance Metrics for Individual Models

Model	Accuracy	Recall	Precision	F1
CNN	0.6762	0.6406	0.6991	0.6657
LSTM	0.7307	0.7146	0.7429	0.7266

4 Results

We trained 10 individual models (5 CNN, 5 LSTM) and then evaluated the effect of different ensembling techniques. For individual scores, we took the model from each class with the highest F1-score as the benchmark for comparison, the key research question then being, can an ensemble model achieve a higher F1-score than any of the individual models could have achieved independently.

Table 3 presents the results for the individual CNN and LSTM architecture.

We then run stacking predictions and log our results in Table 1.

For stacking generalization the results are logged in Table 2.

5 Discussion

5.1 Interpretation of Results

From Table 3 It can be seen that the LSTM outperforms the CNN architecture by about .06 in terms of F1-score.

We observe that for homogeneous architectures, a boost is achieved in both instances as shown in Table 1. CNN achieves a larger boost than LSTM but, as it is also starting at a much lower level, fails to outperform the LSTM architecture. On combining both architectures results in a decrease in performance.

For stacking generalizations, we achieved similar results to stacking predictions but in general managed a smaller boost for ensembling as shown in Table 2. In this case, the combined accuracy of the LSTM ensemble is slightly lower than the best performing individual model and the combined CNN/LSTM ensemble continues to fail to produce a performance improvement.

From this, we conclude that stacking predictions with LSTM exclusively achieves the best performance for the task of Twitter sentiment classification.

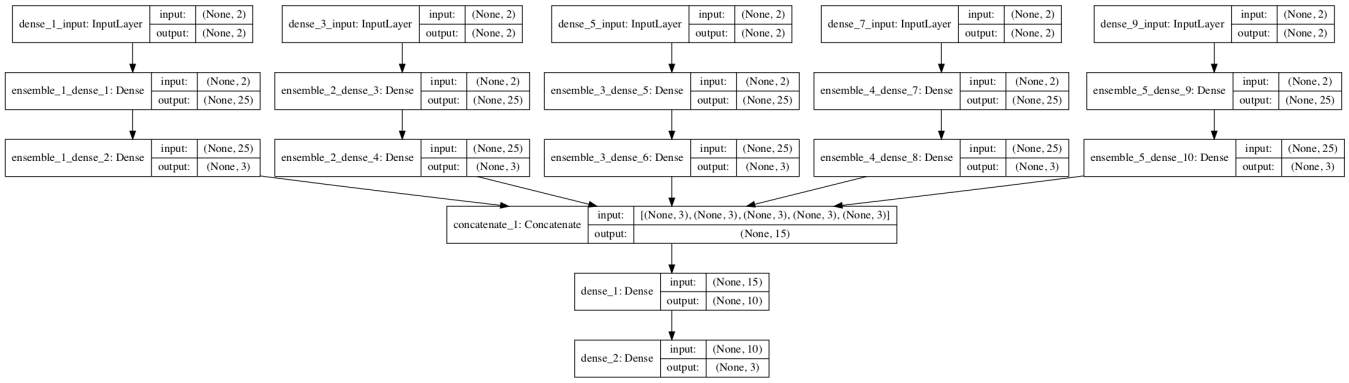


Figure 2: Stacking Generalization Architecture.

5.1.1 Correlation Matrix: In addition to accuracy, we also assessed the correlation between the different models. In general, ensembling techniques should perform better when models have a lower correlation all things being equal.Figure 3) shows the correlation between the five different LSTM models. The key point of variation being the number of LSTM cells in each model. Figure 4 shows the correlation between the 5 different CNN models. The key point of variation is the width of the convolutions in each model.

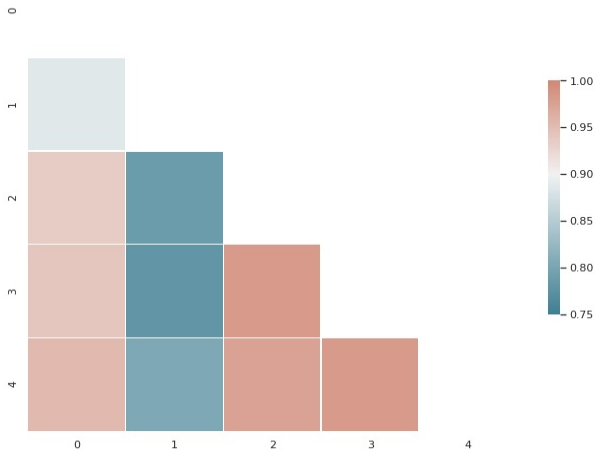


Figure 3: Correlation Matrices for LSTM.

5.1.2 Confusion Matrix for LSTM: We would also like to understand how ensembling approaches tend to compare with a simple averaging of the model predictions in terms of their confusion matrices. This helps to understand how ensembles reclassify their inputs to achieve a better result than the average of their results. In order to do this, we took the average of the LSTM confusion matrices and compared these to the confusion matrices of our ensemble approaches. We focus on the LSTM architecture here since it was the only model where we were able to achieve near to state of the art results.

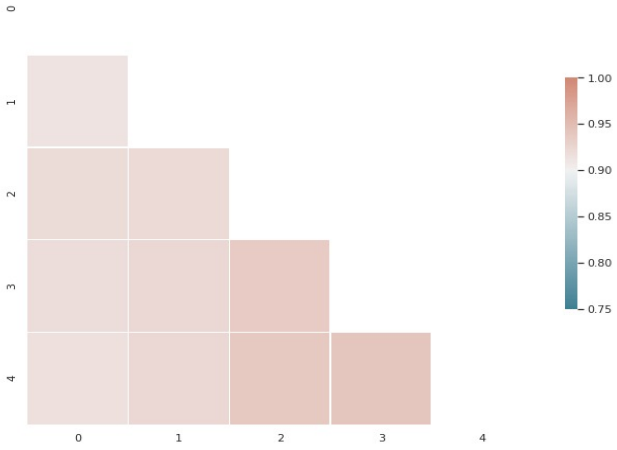


Figure 4: Correlation Matrices for CNN.

Table 4: LSTM Confusion Matrices for Stacking Predictions

		Predicted Labels		
		Neutral	Positive	Negative
Actual Labels	Neutral	81.4	4	-85.4
	Positive	-40	173.4	-133.4
	Negative	4	23.8	27.8

Table 4 shows difference in confusion matrices from the average LSTM model to the ensemble model for stacking prediction(the best performing model). As can be seen, stacking predictions tends to improve performance by reclassifying items out of negative to positive and neutral.

Table 5 shows the difference in confusion matrices from the average LSTM model to the ensemble model for stacking generalization. It can be seen that the reclassifications are a little bit more even and there does not appear to be one class that is more favored.

Table 5: LSTM Confusion Matrices for Stacking Generalization

		Predicted Labels		
		Neutral	Positive	Negative
Actual Labels	Neutral	43.5	2	-45.4
	Positive	-52	117.4	-65.4
	Negative	-18	-5.2	23.2

5.2 The Benefits of Ensembling:

We find in general that using ensembling can sometimes provide a slight boost to neural network performance but in our case that combining a CNN and LSTM architecture does not lead to a synergistic effect but instead that the weaker model tends to drag down the performance of the stronger model.

These results were somewhat surprising since it was assumed that the two models would be able to focus on relatively uncorrelated features and a good ensembling approach could learn when each model's predictions were most appropriate. However, the evidence from our experiments seem to suggest that ensembling will only yield benefits if there are no significant outliers that tend to lower the combined performance. In essence, the team is only as strong as its weakest link.

We also discovered that ensembling seems to have a larger effect when the individual models are simpler and have lower scores individually. The question then becomes when a series of weaker models can outperform a single strong model in a prediction task?

It can be observed in many machine learning tasks that more data is usually better for improving accuracy. All things being equal, a model trained on 1 million examples will outperform one trained on 100K examples. However, there is a point where the marginal increase for additional data becomes less and less. In this experiment, we did not quite hit that upper limit but it is conceivable it could occur in other datasets. In that case, if the performance boost obtained from ensembling can be maintained, training data might be used more profitably to train 10 models with 100K examples each and then ensemble their results to achieve a higher combined accuracy. However, this would only be the case if simply using 1 million examples to train a single neural network only yielded modest improvements since the benefits of ensembling an already strong model can be seen to only achieve a marginal gain.

The second scenario, in the case of the LSTM models we trained, might be if many different architectures yield comparable results to within a few percent of one another and no model is clearly superior. In this case, combining all the approaches might be able to yield a slight performance improvement.

5.3 Precision and Recall:

Seeing the imbalance of precision and recall when using ensembling approaches we also wanted to determine what exactly is happening when we ensemble models. It can be seen that the accuracy of our ensemble model typically manages to obtain a better F1-score by increasing recall over precision.

It seems that what this effectively means is that the class that possesses the most representation in the dataset, i.e. "the majority class", will tend to see an increase in the number of predictions it receives at the expense of the other classes. What is interesting, however, is that different ensembling methods seem to contain different biases. In the case of stacking prediction, there is a clear preference to reclassify items from negative over to positive or neutral, whereas we observe less of a tendency to prioritize negative or neutral in stacking generalization.

It seems that this property might be useful if, for instance, ensembling were to be used in a dataset where it was more critical than certain infrequent classes needed to not be edged out by the majority. However, more research is needed to fully understand the dynamics of how ensembling achieves improved results over a simple averaging of the models but suffice it to say that ensembling and model architectures certainly provide an additional tool to bias a network's errors while maintaining high accuracy.

6 Conclusion

In this paper, we proposed various methods to carry out sentiment analysis using deep learning. The main aim of this work was to conclude whether the ensembling of multiple trained classifiers outperforms individual classifiers. We ended up with 10 intermediate to fairly strong (5 CNN and 5 LSTM) models after carefully tuning different hyper-parameters. These models were finally ensembled using stacking predictions and stacking generalization ensembling techniques. We find that strong ensembling classifiers yield a marginal boost in performance, and having the presence of weaker models in the ensemble stack tends to decrease the overall performance of an ensemble classifier.

As future work, we would like to get more data and have a more even distribution across all classes. We would also like to experiment with other ensembling techniques such as soft/hard voting and Polyak Rupert averaging.

References

[1] Jason Brownlee. 2018. Stacking ensemble for Deep Learning Neural Networks in Python. (2018). https://machinelearningmastery.com/stacking-ensemble-for-deep-learning-neural-networks/

[2] Mathieu Cliche. 2017. Twitter Sentiment Analysis with CNNs and LSTMs. *In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017* (2017). https://doi.org/10.18653/v1/s17-2094

[3] Maha Heikal, Marwan Torki, and Nagwa El-Makky. 2018. Sentiment Analysis of Arabic Tweets using Deep Learning. (November 2018), 114–122.

[4] Yujie Lu, Kotaro Sakamoto, Hideyuki Shibuki, and Tatsunori Mori. 2017. Are Deep Learning Methods Better for Twitter Sentiment Analysis? *In: Proceedings of The Associations for Natural Language Processing* (2017), 789–790.

[5] Pedro M. Sosa. 2017. Twitter Sentiment Analysis using Combine LSTM-CNN Models. (June 2017).

[6] Dario Stojanovski, Gjorgji Strezoski, Gjorgji Madjarov, and Ivica Dimitrovski. 2016. Finki at SemEval-2016 Task 4: Deep Learning Architecture for Twitter Sentiment Analysis. *In: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)* (June 2016), 149–154.

[7] Muhammad Umer, Imran Ashraf, Arif Mehmooda, Saru Kumari, and Gyu Sang Choi Saleem Ullah. 2020. Sentiment Analysis of Tweets using a Unified Convolutional Neural Network-Long Short-Term Memory Network Model. (2020).

[8] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent Neural Network Regularization. (2014). https://arxiv.org/pdf/1409.2329.pdf